Computer vision Seminar

수원대학교

Data Network Analysis 

데이터과학부

정인호

2023.01.26

Week2) Transfer Learning

1. Week1 과제 리뷰
2. Transfer Learning

수원대학교
Data Network Analysis

Week2) Transfer Learning

1. Week1 과제 리뷰

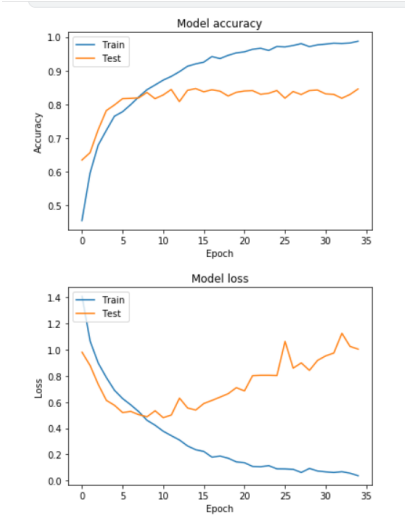
- 과제 피드백
- 1등 코드 리뷰
- 팀장 코드 리뷰

1. week1 과제 리뷰

과제 피드백

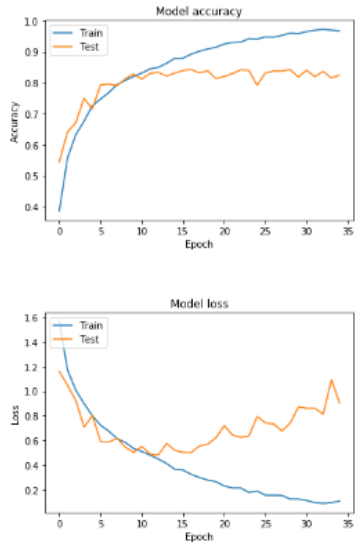
Acc : 0.8413

```
model = Models.Sequential()

model.add(Layers.Conv2D(300,kernel_size=(3,3),activation='relu',input_shape=(150,15
model.add(Layers.Conv2D(300,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(4,4))
model.add(Layers.Conv2D(256,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(240,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(128,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(4,4))
model.add(Layers.Flatten())
model.add(Layers.Dense(256,activation='relu'))
model.add(Layers.Dense(128,activation='relu'))
model.add(Layers.Dense(64,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossent

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```



0.8426 원본 코드

```
model = Models.Sequential()

model.add(Layers.Conv2D(200,kernel_size=(3,3),activation='relu',input_shape=(150,150,3)))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(100,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Flatten())
model.add(Layers.Dense(180,activation='relu'))
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```



https://www.kaggle.com/code/uzairrj/beg-tut-intel-image-classification-93-76-accur/notebook

1. week1 과제 리뷰

과제 피드백

Acc : 0.8457 – 서X준

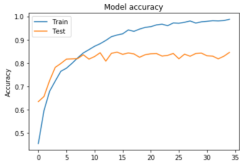
```
model.add(Layers.Conv2D(64,kernel_size=(3,3),activation='relu',padding = 'same',input_shape=(224,224,3), kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.BatchNormalization())
# (none, 30,30,64)

model.add(Layers.Conv2D(128,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.BatchNormalization())
# (none, 15,15,128)

model.add(Layers.Conv2D(256,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(256,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.BatchNormalization())
# (none, 5,5,256)


model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.BatchNormalization())
# (none, 1,1,512 )

model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(512,kernel_size=(3,3),activation='relu',padding = 'same', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.MaxPool2D(2,2))
model.add(Layers.BatchNormalization())


model.add(Layers.Flatten())
# 512


model.add(Layers.Dense(4096,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(4096,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(1000,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(600,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```

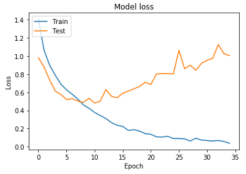
모델 상세 비교

서X준

Layer (type)	Output Shape	Param #
batch_normalization_v1_12 (B	(None, 28, 28, 512)	2048
max_pooling2d_3 (MaxPooling2	(None, 14, 14, 512)	0
batch_normalization_v1_13 (B	(None, 14, 14, 512)	2048
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2359808
batch_normalization_v1_14 (B	(None, 14, 14, 512)	2048
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
batch_normalization_v1_15 (B	(None, 14, 14, 512)	2048
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
batch_normalization_v1_16 (B	(None, 14, 14, 512)	2048
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
batch_normalization_v1_17 (B	(None, 14, 14, 512)	2048
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 512)	0
batch_normalization_v1_18 (B	(None, 7, 7, 512)	2048
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102764544
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 1000)	4097000
dropout_2 (Dropout)	(None, 1000)	0
dense_3 (Dense)	(None, 600)	600600
dropout_3 (Dropout)	(None, 600)	0
dense_4 (Dense)	(None, 6)	3606

Total params: 143,117,142
Trainable params: 143,104,214
Non-trainable params: 12,928

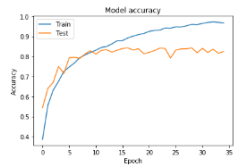


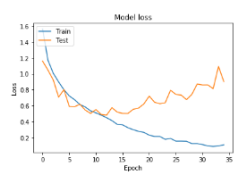


원본코드

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 200)	5600
conv2d_1 (Conv2D)	(None, 146, 146, 180)	324180
max_pooling2d (MaxPooling2D)	(None, 29, 29, 180)	0
conv2d_2 (Conv2D)	(None, 27, 27, 180)	291780
conv2d_3 (Conv2D)	(None, 25, 25, 140)	226940
conv2d_4 (Conv2D)	(None, 23, 23, 100)	126100
conv2d_5 (Conv2D)	(None, 21, 21, 50)	45050
max_pooling2d_1 (MaxPooling2	(None, 4, 4, 50)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 180)	144180
dense_1 (Dense)	(None, 100)	18100
dense_2 (Dense)	(None, 50)	5050
dropout (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 6)	306

Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0





https://www.kaggle.com/code/uzairrj/beg-tut-intel-image-classification-93-76-accur/notebook

1. week1 과제 리뷰

과제 피드백

Acc : 0.1710 (???)

```
model = Models.Sequential()

model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu', padding = 'same', input_shape=(150,150,3), kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(80, kernel_size=(3,3), activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.MaxPool2D(5,5))

model.add(Layers.Conv2D(80,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(110,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.MaxPool2D(5,5))


model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(170,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(190,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(220,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.Conv2D(250,kernel_size=(3,3),activation='relu', padding = 'same', kernel_initializer = 'he_normal'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Flatten())
model.add(Layers.Dense(180,activation='relu'))
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dense(25,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```

[9] :

```
trained = model.fit(Images,Labels,epochs=15,validation_split=0.20)
```

```
Train on 11227 samples, validate on 2807 samples
Epoch 1/15
11227/11227 [==============================] - 23s 2ms/sample - loss: 13.3536 - acc: 0.1705 - val_loss: 13.5284 - val_acc: 0.1607
Epoch 2/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.2718 - acc: 0.1762 - val_loss: 13.5284 - val_acc: 0.1607
Epoch 3/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.2446 - acc: 0.1780 - val_loss: 13.5284 - val_acc: 0.1607
Epoch 4/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.4499 - acc: 0.1654 - val_loss: 13.5284 - val_acc: 0.1607
Epoch 5/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.3424 - acc: 0.1721 - val_loss: 13.5284 - val_acc: 0.1607
Epoch 6/15
11227/11227 [==============================] - 20s 2ms/sample - loss: 13.4116 - acc: 0.1677 - val_loss: 13.3619 - val_acc: 0.1710
Epoch 7/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.3972 - acc: 0.1683 - val_loss: 13.3619 - val_acc: 0.1710
Epoch 8/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.5216 - acc: 0.1608 - val_loss: 13.3619 - val_acc: 0.1710
Epoch 9/15
11227/11227 [==============================] - 20s 2ms/sample - loss: 13.4325 - acc: 0.1664 - val_loss: 13.4940 - val_acc: 0.1628
Epoch 10/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.4036 - acc: 0.1681 - val_loss: 13.4940 - val_acc: 0.1628
Epoch 11/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.3862 - acc: 0.1690 - val_loss: 13.4940 - val_acc: 0.1628
Epoch 12/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.3858 - acc: 0.1685 - val_loss: 13.3619 - val_acc: 0.1710
Epoch 13/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.4465 - acc: 0.1650 - val_loss: 13.3619 - val_acc: 0.1710
Epoch 14/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.3987 - acc: 0.1671 - val_loss: 13.4940 - val_acc: 0.1628
Epoch 15/15
11227/11227 [==============================] - 19s 2ms/sample - loss: 13.4310 - acc: 0.1636 - val_loss: 13.4940 - val_acc: 0.1628
```

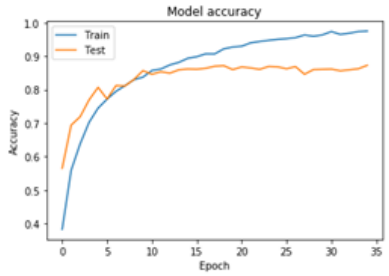
https://www.kaggle.com/code/uzairrj/beg-tut-intel-image-classification-93-76-accur/notebook

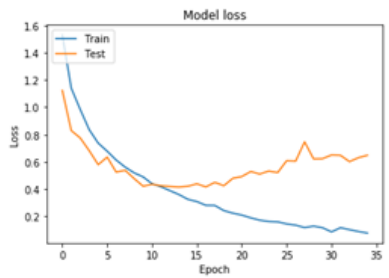
1. week1 과제 리뷰

과제 피드백

1등 : 0.8687 – 구성준

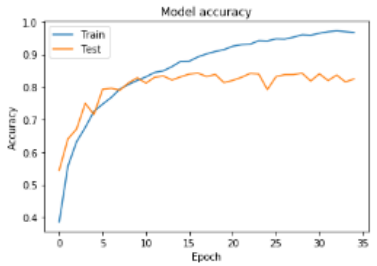
```
model = Models.Sequential()

model.add(Layers.Conv2D(256,kernel_size=(3,3),activation='relu',padding = 'same',input_shape=(150,150,3)))
model.add(Layers.Conv2D(256,kernel_size=(3,3),padding = 'same',activation='relu'))
model.add(Layers.MaxPool2D(5,5))

model.add(Layers.Conv2D(128,kernel_size=(3,3),activation='relu',padding = 'same'))
model.add(Layers.Conv2D(128,kernel_size=(3,3),activation='relu',padding = 'same'))
model.add(Layers.MaxPool2D(5,5))

model.add(Layers.Conv2D(64,kernel_size=(3,3),activation='relu',padding = 'same'))
model.add(Layers.Conv2D(64,kernel_size=(3,3),activation='relu',padding = 'same'))
model.add(Layers.MaxPool2D(5,5))

model.add(Layers.Flatten())
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```

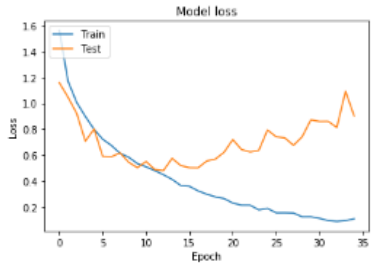




0.8426 원본 코드

```
model = Models.Sequential()

model.add(Layers.Conv2D(200,kernel_size=(3,3),activation='relu',input_shape=(150,150,3)))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(100,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Flatten())
model.add(Layers.Dense(180,activation='relu'))
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
SVG(model_to_dot(model).create(prog='dot', format='svg'))
Utils.plot_model(model,to_file='model.png',show_shapes=True)
```





1. week1 과제 리뷰

과제 피드백

발표자 본인 코드 0.89

```
model = Models.Sequential()

model.add(Layers.Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',input_shape=(150,150,3), kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(32,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.GlobalAveragePooling2D())
model.add(Layers.Dense(200, activation='relu', kernel_initializer='he_normal'))
model.add(Layers.Dropout(rate=0.8))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0005),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

원본 코드와 파라미터 비교

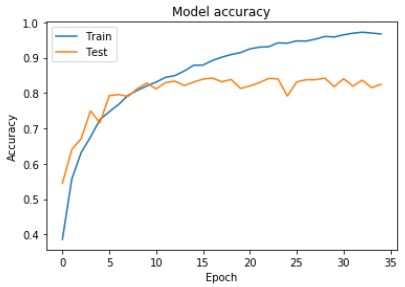
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	896
batch_normalization_v1 (Batc	(None, 150, 150, 32)	128
conv2d_1 (Conv2D)	(None, 150, 150, 32)	9248
batch_normalization_v1_1 (Ba	(None, 150, 150, 32)	128
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_2 (Conv2D)	(None, 50, 50, 64)	18496
batch_normalization_v1_2 (Ba	(None, 50, 50, 64)	256
conv2d_3 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_3 (Ba	(None, 50, 50, 64)	256
conv2d_4 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_4 (Ba	(None, 50, 50, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_v1_5 (Ba	(None, 16, 16, 128)	512
conv2d_6 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_6 (Ba	(None, 16, 16, 128)	512
conv2d_7 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_7 (Ba	(None, 16, 16, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 128)	0
global_average_pooling2d (Gl	(None, 128)	0
dense (Dense)	(None, 200)	25800
dropout (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 6)	1206

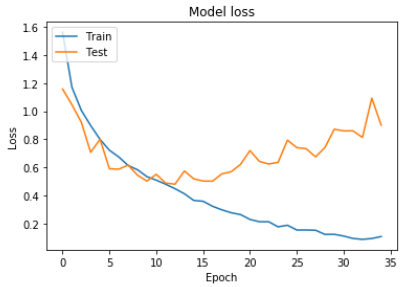
Total params: 501,086
Trainable params: 499,806
Non-trainable params: 1,280

flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 180)	144180
dense_1 (Dense)	(None, 100)	18100
dense_2 (Dense)	(None, 50)	5050
dropout (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 6)	306

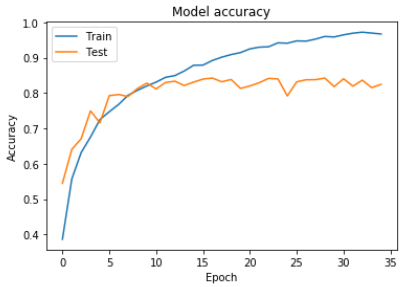
Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0

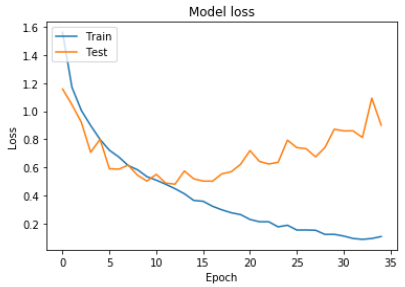
발표자





원본 코드





블로그 관련 포스팅: https://inhovation97.tistory.com/43

1. week1 과제 리뷰

지난 시간 배운 CNN을 복기해 봅시다.

발표자 본인 코드 0.89

```
model = Models.Sequential()

model.add(Layers.Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',input_shape=(150,150,3), kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(32,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.GlobalAveragePooling2D())
model.add(Layers.Dense(200, activation='relu', kernel_initializer='he_normal'))
model.add(Layers.Dropout(rate=0.8))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0005),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

원본 코드와 파라미터 비교

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 180)	144180
dense_1 (Dense)	(None, 100)	18100
dense_2 (Dense)	(None, 50)	5050
dropout (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 6)	306

Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	896
batch_normalization_v1 (Batc	(None, 150, 150, 32)	128
conv2d_1 (Conv2D)	(None, 150, 150, 32)	9248
batch_normalization_v1_1 (Ba	(None, 150, 150, 32)	128
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_2 (Conv2D)	(None, 50, 50, 64)	18496
batch_normalization_v1_2 (Ba	(None, 50, 50, 64)	256
conv2d_3 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_3 (Ba	(None, 50, 50, 64)	256
conv2d_4 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_4 (Ba	(None, 50, 50, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_v1_5 (Ba	(None, 16, 16, 128)	512
conv2d_6 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_6 (Ba	(None, 16, 16, 128)	512
conv2d_7 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_7 (Ba	(None, 16, 16, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 128)	0
global_average_pooling2d (Gl	(None, 128)	0
dense (Dense)	(None, 200)	25800
dropout (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 6)	1206

Total params: 501,086
Trainable params: 499,806
Non-trainable params: 1,280

Learning Rate & Batch size의 상관 관계: https://inhovation97.tistory.com/32

1. week1 과제 리뷰

지난 시간 배운 CNN을 복기해 봅시다.

발표자 본인 코드 0.89

```
model = Models.Sequential()

model.add(Layers.Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',input_shape=(150,150,3), kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(32,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(64,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())
model.add(Layers.Conv2D(128,kernel_size=(3,3),padding='same',activation='relu', kernel_initializer='he_normal'))
model.add(Layers.BatchNormalization())

model.add(Layers.MaxPool2D(3,3))

model.add(Layers.GlobalAveragePooling2D())
model.add(Layers.Dense(200, activation='relu', kernel_initializer='he_normal'))
model.add(Layers.Dropout(rate=0.8))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0005),loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

원본 코드와 파라미터 비교

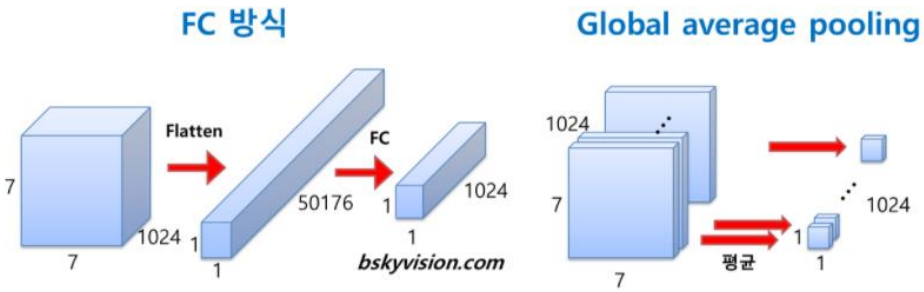
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	896
batch_normalization_v1 (Batc	(None, 150, 150, 32)	128
conv2d_1 (Conv2D)	(None, 150, 150, 32)	9248
batch_normalization_v1_1 (Ba	(None, 150, 150, 32)	128
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_2 (Conv2D)	(None, 50, 50, 64)	18496
batch_normalization_v1_2 (Ba	(None, 50, 50, 64)	256
conv2d_3 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_3 (Ba	(None, 50, 50, 64)	256
conv2d_4 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_v1_4 (Ba	(None, 50, 50, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_v1_5 (Ba	(None, 16, 16, 128)	512
conv2d_6 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_6 (Ba	(None, 16, 16, 128)	512
conv2d_7 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_v1_7 (Ba	(None, 16, 16, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 128)	0
global_average_pooling2d (Gl	(None, 128)	0
dense (Dense)	(None, 200)	25800
dropout (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 6)	1206

Total params: 501,086
Trainable params: 499,806
Non-trainable params: 1,280

flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 180)	144180
dense_1 (Dense)	(None, 100)	18100
dense_2 (Dense)	(None, 50)	5050
dropout (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 6)	306

Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0

Global Average Pooling



만약 FC 방식을 사용한다면 훈련이 필요한
가중치의 개수가 7 x 7 x 1024 x 1024 = 51.3M이지만

global average pooling을 사용하면 가중치가 단 한개도 필요하지 않음.

1. week1 과제 리뷰

정리하기

1. 기울기 소실 문제

2. 과적합 방지

3. Global Average Pooling

4. Batchsize & Learning rate

Week2) Transfer Learning

2. Transfer Learning

- Imagenet Chanllenge
- Transfer Learning

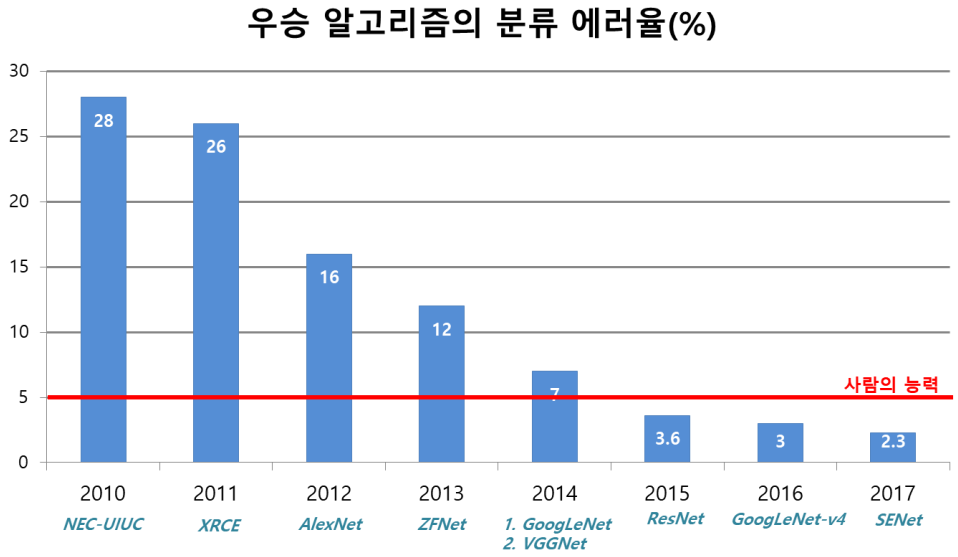
2. Transfer Learning

Imagenet challenge

이미지넷 챌린지

Train set : 130만개
Validation set : 5만개
Test set : 10만개
138GB가 넘는 데이터 셋

***ILSVRC 2012 데이터 셋*으로 1,000개의 클래스를 분류하는 국제 대회**



이 대회에서 2011년까지는 이미지 인식률이 75%를 넘지 못하였는데, 2012년 대회에서 인공신경망을 이용한 알렉스넷(AlexNet)이 무려 84.7%라는 놀라운 인식률을 달성합니다. 그 이후부터는 딥러닝을 이용한 인공지능이 상위 랭크를 모두 휩쓸었고, 매년 인식률이 높아져, 현재는 상당수의 도전자가 97%에 육박하는 인식률을 기록하고 있습니다. 이는 인간의 인식률인 95%를 훨씬 웃도는 수준입니다.

https://wikidocs.net/28924

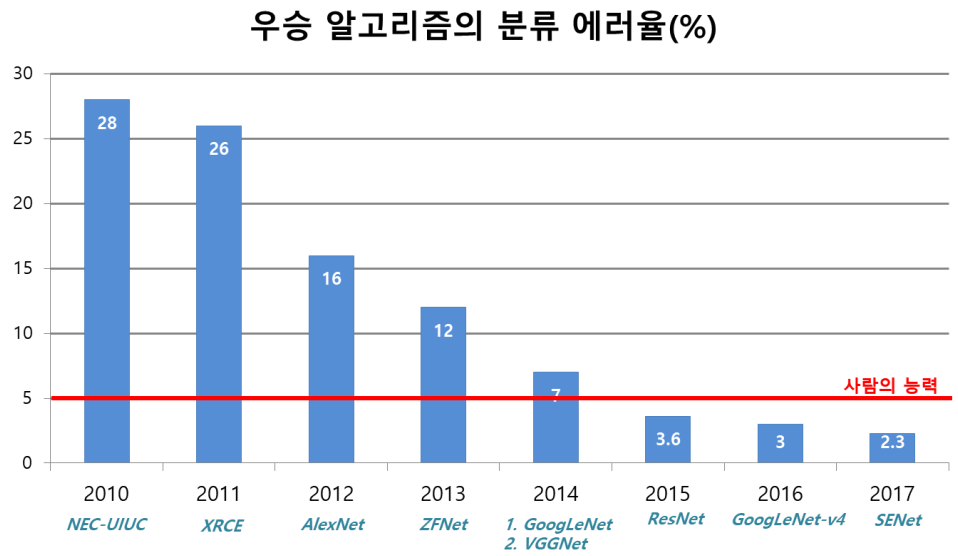
2. Transfer Learning

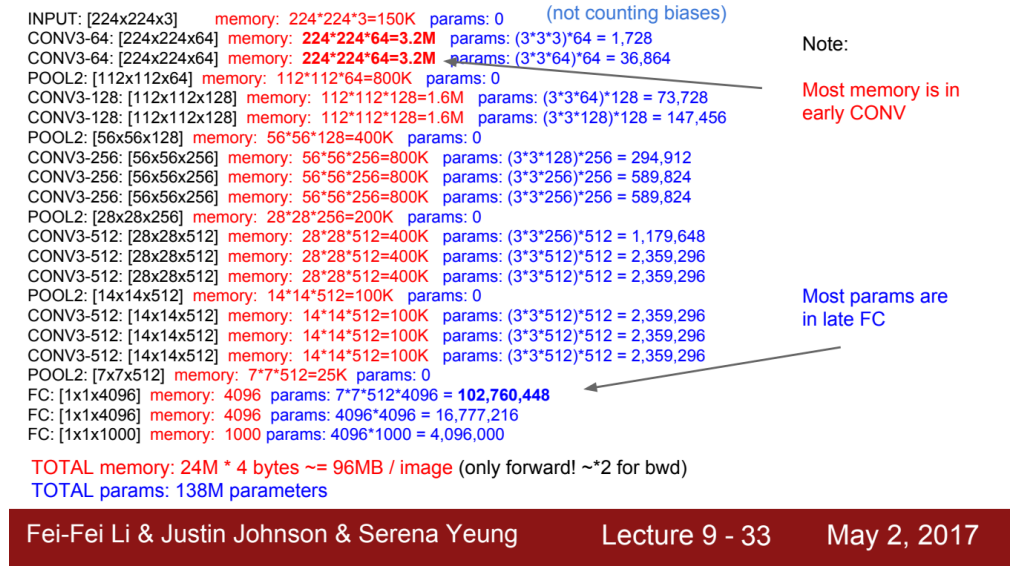
Imagenet challenge

Week1의 과제에서는 학습 이미지 셋은 14,034 여 장

사용자가 가지고 있는 이미지는 각각 다르다.

어떻게 깊은 신경망의 Sotar인 모델을 내가 가진 데이터 셋에 맞게 이용할 수 있을까?





2. Transfer Learning

Finetuning

소규모 데이터 셋으로 좋은 모델을 쓰고 싶다.

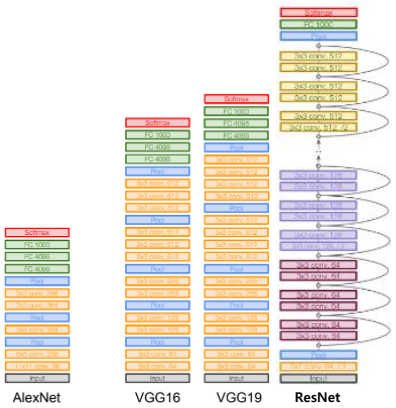
: Train (LR = original LR / 10)

: Frozen (LR = 0)

Transfer Learning(전이 학습)

Fine tuning

Backbone



2. Transfer Learning

Featrue Extraction from Pretrained model

1. 합성곱 층에 의해 학습된 표현이 더 일반적이어서 재사용 가능하다.
2. ConvNet의 Feature map은 이미지에 대한 일반적인 컨셉의 존재 여부를 기록한 map이다.
3. 하지만 분류기(Fc Layer)에서 학습한 표현은 모델이 훈련된 클래스 집합에 특화되어 있습니다.
4. 분류기는 전체 사진에 어떤 클래스가 존재할 확률에 관한 정보만을 담고 있습니다.
5. 더군다나 완전 연결 층(Fc Layer)에서 찾은 표현은 더 이상 입력 이미지에 있는 객체의 위치 정보를 가지고 있지 않습니다.
6. 완전 연결 층들은 공간 개념을 제거하지마 합성곱의 feature map은 객체의 위치를 고려합니다. 객체의 위치가 중요한 문제라면 완전 연결 층(Fc Layer)에서 만든 특성은 크게 쓸모가 없습니다.

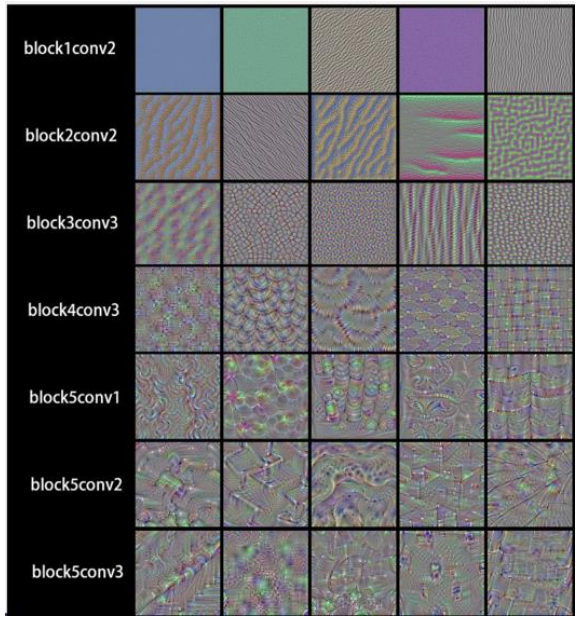
2. Transfer Learning

Featrue Extraction from Pretrained model

1. 특정 합성곱 층에서 추출한 표현의 일반성 (그리고 재사용성)의 수준은 모델에 있는 층의 깊이에 달려 있습니다.

2. 모델의 하위 층은 (에지, 색깔, 질감 등과 같이) 지역적이고 매우 일반적인 특성 맵을 추출합니다. 반면 상위 층은 (강아지 눈 or 고양이 귀와 같이) 좀 더 추상적인 개념을 추출합니다. Local -> Global

3. 만약 새로운 데이터 셋이 원본 모델이 훈련한 데이터 셋과 많이 다르다면, 전체 합성곱 기반층을 사용하는 것 보다는 모델의 하위 층 몇 개만 특성 추출에 사용하는 것이 좋습니다.



Layer가 깊어질수록 filter는 더 큰 특징과 추상적인 Feature들을 가져간다.

2. Transfer Learning

week2
과제 설명

이미지 디렉토리 관리하기